COBB

# INSIDE DOS™

*Tips & techniques for MS-DOS & PC-DOS*

*March 1992*
*Vol. 3 No. 3*

## Two ways to shorten your command path

VERSIONS 3.1-5.0

*Contributing Editor Tim Landgrave provided us with much of the information in this article.*

If your hard disk is organized into several levels of directories, you probably find yourself typing very long pathnames as you issue commands from the DOS prompt or as you work with data files within your applications. For example, if you store your documents in the directory

```
C:\APPS\WP\LETTERS
```

you'll often find yourself typing in that long pathname whenever you need to access any of the files in that directory.

Also, if you routinely work in several subdirectories, you might have been frustrated by the path's length limitations. The path is limited by the length of an environment string, which is 127 characters. When you deduct the five characters in *path*=, you can only specify a path of 122 characters.

In this article, we'll present a couple of tricks you can use to get around these problems. Both methods take advantage of the SUBST command. First, we'll introduce you to the SUBST command, and then we'll demonstrate a couple of situations where it can really save you keystrokes.

### The SUBST command

The SUBST command lets you assign a drive letter to any directory. Once you've used the SUBST command to assign a drive letter to a directory, you can treat that directory as if it were a separate drive.

The basic form of the SUBST command is

```
subst drive path
```

where *drive* is the drive letter (followed by a colon) you want to assign, and *path* is the full pathname of the directory to which you want *drive* to refer. Once you've used a SUBST command to assign a drive to a path, you can use a command in the form

```
subst drive /d
```

to cancel the assignment, where *drive* is the drive letter you want to remove.

### An example

For example, suppose you often refer to the files stored in the directory C:\APPS\WP\LETTERS, and you want to treat that directory as if it were a separate drive E. To do this, simply issue the command

```
C:\>subst e: c:\apps\wp\letters
```

Once you've issued this command, you can work with the files in C:\APPS\WP\LETTERS much more easily than you could before. For instance, to display a listing of all the files stored in that directory, you can simply use the command

```
C:\>dir e:
```

instead of the command

```
C:\>dir c:\apps\wp\letters
```

Similarly, if you want to copy the file LANE.DOC from the directory C:\APPS\WP\LETTERS to drive A, you can use the command

```
C:\>copy e:lane.doc a:
```

instead of the command

```
C:\>copy c:\apps\wp\letters\lane.doc a:
```

If you later decide you want to remove the letter E dsignation from the directory C:\APPS\WP\LETTERS, you'll need to issue the command

```
C:\>subst e: /d
```

*Continued on page 11*

**IN THIS ISSUE**

*A Publication of The Cobb Group*

# INSIDE DOS

*Inside DOS* (ISSN 1049-5320) is published monthly by The Cobb Group.

| | |
|---|---|
| **Prices** | Domestic .................... $59/yr. ($7.00 each)<br>Outside the US ........... $79/yr. ($8.50 each) |
| **Address** | The Cobb Group<br>9420 Bunsen Parkway, Suite 300<br>Louisville, KY 40220 |
| **Phone** | Toll free .......... (800) 223-8720<br>Local ............. (502) 491-1900<br>FAX ............... (502) 491-8050 |

| | | |
|---|---|---|
| **Staff** | Editor-in-Chief | Jim Welp |
| | Consulting Editors | Mark W. Crane |
| | | Tim Landgrave |
| | Contributing Editor | Van Wolverton |
| | Editing | Clyde Zellers |
| | | Polly Blakemore |
| | Production | Debbie Lane |
| | Design | Karl Feige |
| | Publications Manager | Elayne Noltemeyer |
| | Publisher | Douglas Cobb |

Address correspondence and special requests to The Editor, *Inside DOS*, 9420 Bunsen Parkway, Suite 300, Louisville, KY 40220. Address subscriptions, fulfillment questions, and requests for bulk orders to Customer Relations, at the same address.

**Postmaster:** Send address changes to *Inside DOS*, P.O. Box 35160, Louisville, KY 40232. Second class postage is paid in Louisville, KY.

## Conventions

To avoid confusion, we would like to explain a few of the conventions used in *Inside DOS*.

When we instruct you to type something, those characters usually appear on a separate line along with the DOS prompt. The characters you type will appear in red, while the characters DOS displays will appear in black.

Occasionally, we won't display the characters you type on a separate line. In these cases, we'll display the characters you type in italics. For example, we might say, "Issue the command *dir *.txt* at the DOS prompt." Although DOS is not case-sensitive, we'll always display the characters you type in lowercase.

When we refer to a general DOS command (not the command you actually type at the DOS prompt), we'll display that command name in all caps. For example, we might say, "You can use either the COPY or XCOPY command to transfer files from one disk to another."

Many commands accept parameters that specify a particular file, disk drive, or other option. When we show you the form of such a command, its parameters will appear in italics. For example, the form of the COPY command is

```
copy file1 file2
```

where *file1* and *file2* represent the names of the source file and the target file, respectively.

The names of keys, such as [Shift], [Ctrl], and [F1], appear in brackets. When two keys must be pressed simultaneously, those key names appear side by side, as in [Ctrl][Break] or [Ctrl]z.

## LETTERS

### A better way to avoid the DEL command's verification prompt

VERSIONS 2.1-5.0

I'm writing in response to a letter that appeared in your December 1991 issue in which a teacher asked how to avoid the verification prompt when he typed

```
A:\>del *.*
```

You suggested that he create a text file called Y.TXT with nothing more than the character Y in it. Then, you created a batch file called DELALL.BAT, in which you redirected Y.TXT to the *del *.** command. Doing so causes DOS to supply the Y response to the prompt:

```
All files in directory will be deleted!
Are you sure (Y/N)?
```

Although this technique works, it takes up a lot of disk space because it requires two files. A better way of avoiding the verification prompt is to take advantage of the pipe character (¦). Using the pipe symbol, you can redirect the character Y with only the command

```
A:\>echo y ¦ del *.*
```

As you suggested in the December issue, you can create a batch file that contains this command. However,

**See Microsoft's suite of Windows applications—including the new Word for Windows 2.0—at a FREE Microsoft seminar**

Microsoft invites you to discover firsthand how to get your work done faster and easier with its suite of Windows applications, including the new Microsoft Word for Windows 2.0. By attending one of Microsoft's free seminars, you can see Word for Windows in action, along with Microsoft Excel, Microsoft Powerpoint, and Microsoft Project. Call (800) 227-4679 and ask for Dept. CD32 to find out when and where a seminar is coming to your area.

if you're short on disk space, you can simply type the command, then easily recall it with the [F3] key each time you want to repeat it. I hope other *Inside DOS* subscribers find this tip useful.

*Matthew Russell*
*East Islip, New York*

Indeed it is, Mr. Russell; and you were not alone in pointing out this simpler method. Several readers suggested using the ECHO command and the pipe redirection symbol. Because it doesn't require a separate text file, this command is more accessible and requires less disk space.

A few of you also recommended some other enhancements to DELALL.BAT. One safeguard is to include the drive specification after the DEL command:

```
echo y ¦ del A: *.*
```

When you specify the A drive, you won't run the risk of accidentally deleting files on your hard drive. Of course, if you need to delete files on another drive, you'd need to change the drive specification.

## A new and improved DELALL.BAT

Subscriber James S. Neiman, Sr., pointed out that typing *delall* each time you insert a new diskette isn't the most convenient way to delete the files on hundreds of diskettes. He suggested a batch file that prompts you to insert diskettes and continue by pressing any key. Figure A shows a new and improved DELALL.BAT.

Let's take a quick look at how it works. Before we do, though, we'll repeat the warning we gave in the December issue: The DEL command is dangerous enough without overriding its built-in warning prompt. When you create and run this batch file, be very careful or you could mistakenly delete the wrong files.

When you run DELALL.BAT, the first command, *@echo off,* tells DOS not to display the commands as it carries them out. That command is followed by the label :NEXT, which, along with the GOTO command in the last line, tells DOS to loop back to the beginning of the batch file. These two commands allow the batch file to delete multiple diskettes.

The CLS command in the third line clears the screen, and the ECHO command in the next line prompts you to insert the diskette. The PAUSE command then temporarily stops the execution of the batch file until you press any key. The combination of the ECHO command and the PAUSE command causes DOS to display the message

```
Place diskette to be deleted in drive A:
Press any key to continue . . .
```

(By the way, if you're using a version of DOS earlier than version 4, the PAUSE command displays the message *Strike a key when ready . . . .*) The next command

```
echo y ¦ del a:*.*
```

is the heart of the batch file. As we mentioned earlier, the *echo y ¦* portion of the command supplies the Y response to the DEL command's verification prompt.

The next command provides the option of ending the batch file's execution by pressing [Ctrl][Break] or pressing any key to delete another diskette. Actually, this is a phantom command because pressing [Ctrl][Break] will end the execution of any batch file, and the function of pressing any key to continue is actually carried out by the next command:

```
pause > nul
```

This command tells DOS to pause just as it did before, only without supplying the message

```
Press any key to continue . . .
```

Finally, the command

```
goto NEXT
```

tells DOS to loop to the label :NEXT at the beginning of the batch file and start the process over again.

**Figure A**

```
@echo off
:NEXT
cls
echo Place diskette to be deleted in drive A:
pause
echo y ¦ del a:*.*
echo Press Ctrl-Break to quit; press any other key to delete another diskette.
pause > nul
goto NEXT
```

*This new and improved DELALL.BAT deletes all the files in the root directory of the diskette in drive A and prompts the user to insert another diskette.*

## Running DELALL.BAT

To run DELALL.BAT, simply type

```
C:\>delall
```

When you do, DOS will prompt you to insert into drive A the diskette whose files you want to delete and to press any key to continue. Then it will delete all the files on the diskette and present the prompt

```
Press Ctrl-Break to quit; press any other key to
delete another diskette.
```

If you want to end the batch file, press [Ctrl][Break] and DOS will present the prompt

```
Terminate batch job (Y/N)?
```

Press *Y* to return to the prompt. If you want to delete another diskette, just press any key and DOS will carry out the GOTO command, looping to the beginning of the batch file, clearing the screen, and prompting for another diskette with the message

```
Place diskette to be deleted in drive A:
Press any key to continue . . .
```

As you can see, creating the improved DELALL.BAT greatly reduces the work involved in deleting all the files on hundreds of diskettes.

EXTENDING DOS

VERSIONS 2.1-5.0

# Soliciting a reply

**By Van Wolverton**

One of the more popular types of DOS-enhancing programs is one that lets a batch file accept user input. Articles in the September and December 1990 issues of *Inside DOS* described two versions of a program named KEYPRESS.COM that wait for you to press a key, then set the DOS ERRORLEVEL value according to the key you pressed. The simpler version described in the September issue set the ERRORLEVEL to 3 for Y, 2 for N, or 1 for [Esc], letting you use prompts in a batch file that could be answered with *Yes*, *No*, or *Escape* for quit. The more elaborate version described in the December issue let you display a prompt message without using an ECHO command, if you wished, and accepted any alphabetic key (A-Z) or [Esc].

Here's another variation on that theme. This program, called REPLY.COM, accepts any key or valid key combination (such as [Ctrl]J or [Alt][F10]), and sets ERRORLEVEL to the key code of the key that was pressed (the extended key code of combinations). REPLY.COM is pretty simple: It doesn't display a prompt, and there is some duplication in the ERRORLEVEL values it produces, but it has the virtue of being quite short (only 14 bytes).

The routine for creating the program is the same as the one used to create KEYPRESS.COM: First you create a script file that contains commands for the DOS DEBUG program, then you start DEBUG and redirect its input to the script file. When DEBUG carries out the commands in the script file, it creates the file named REPLY.COM that contains the program and returns to DOS.

(By the way, REPLY.COM comes from my book *Supercharging MS-DOS*, published by Microsoft Press, which shows a number of techniques for writing batch files and using other advanced DOS features.)

## Creating the script file

A DEBUG script file must contain only ASCII characters. You can create REPLY.SCR with a text editor or word processor that lets you store a text file with no special formatting codes. (Microsoft Word calls this the *text-only* option on the Transfer menu.) If you're not sure whether your word processor lets you do this, you can use Edlin, the text editor that comes with DOS; if you have version 5 of DOS, you can use the DOS Editor, which is much easier to use than Edlin.

The quickest way to create REPLY.SCR is simply to copy the lines from the console to the file. Once you've entered a line, however, you can't go back and change it, so make sure each line is correct before you press [Enter]. Type the following lines exactly as shown (if you use a text editor or word processor, omit the first and last lines):

```
C:\>copy con reply.scr
e100 b4 08 cd 21 3c 00 75 02 cd 21 b4 4c cd 21
rcx
e
w
q
[F6][Enter]
```

[F6] in the last line means press the function key labeled F6, then press [Enter]. DOS responds *1 file(s) copied*.

If you use a text editor or word processor to create REPLY.SCR, double-check the contents of the file against the preceding instructions, then save it with the name REPLY.SCR.

## Creating REPLY.COM

Creating REPLY.COM from the script file takes you even less time. A single command starts DEBUG, names the file you're going to create (REPLY.COM) and tells DOS to redirect its input from the keyboard to the file named REPLY.SCR. Type the following:

```
C:\>debug reply.com < reply.scr
```

DEBUG should create REPLY.COM, then return to the DOS system prompt. If you check with the DIR command, you should see the entry for REPLY.COM:

```
REPLY    COM        14   3-15-91   7:34p
```

If DOS doesn't display an entry for REPLY.COM, or if the file isn't 14 bytes long, edit REPLY.SCR and make sure it contains exactly the lines shown above (or re-create it) and enter the command to start DEBUG again.

## Testing REPLY.COM

It's easy to test REPLY.COM: Just type *reply*. The cursor should sit at the beginning of the line below the system prompt, waiting for you to press a key. Press any key, and DOS should display the system prompt. This doesn't tell you much, of course, other than the fact that the program runs without locking up your system (if it does lock up your system, restart and create it again).

To make sure REPLY.COM is working correctly (and to see it do something useful), you must use it in a batch file. Type the following to create a simple test file named TEST.BAT (if you use a text editor or word processor, omit the first and last lines):

```
C:\>copy con test.bat
@echo off
echo Press Ctrl-K
:WAIT
reply
if errorlevel 11 if not errorlevel 12 goto END
goto WAIT
:END
[F6][Enter]
```

This batch file displays the message *Press Ctrl-K* and waits for you to press a key. If you press Ctrl-K, it returns to DOS; if you press any other key, it continues to wait. The key code for Ctrl-K is 11, so the IF ERRORLEVEL command makes sure that the ERRORLEVEL returned by REPLY.COM is 11 or more but not 12 or more; only 11 satisfies those two conditions.

For a more useful application of REPLY.COM, see the following article "Teaching an Old DIR New Tricks," which describes a batch file that creates a menu of options for DOS 5's DIR command.

VAN WOLVERTON

VERSION 5.0

# Teaching an old DIR new tricks

**By Van Wolverton**

Most of the press coverage of DOS 5 has gone to sexy new features such as memory management and UNDELETE, but some of the old reliable DOS commands have been gussied up, too. A few jewels are lurking, in fact, among the changes made to familiar commands; the Directory command (more commonly called DIR), in particular, has become much more versatile with its new DOS 5 capabilities.

The DOS Shell, first introduced with version 4, is Microsoft's response to third-party shell programs that simplify file management. Windows, too, owes its existence in part to the desire to take the drudgery out of finding, copying, deleting, and otherwise keeping track of the hundreds or thousands of files that can accumulate on a hard disk.

Running DOS from the command line can never offer the convenience of these other programs, but the changes to DIR in version 5 let you streamline your housekeeping chores.

## Showing your files in order

Probably the most talked-about addition to DIR is its ability to specify the order in which it displays directory entries. When DOS utility programs first appeared, control over the order of the directory entries was offered as a genuine improvement. Now you can do it with DOS itself, using the /O (for *order*) parameter.

You can order the list by any of the four categories governing each directory entry, as shown in Table A.

**Table A**

| Option | Entries ordered by |
|---|---|
| /ON | Name |
| /OE | Extension |
| /OS | Size |
| /OD | Date and time |

You can also display all the subdirectories at the beginning of the list by specifying /OG (for *group*).

To display the directory entries ordered by name, for example, you would type

```
C:\>dir /on
```

To display them ordered by extension, you would type

```
C:\>dir /oe
```

You can combine options. To display the directory entries ordered by name with the subdirectories at the beginning, you would type

```
C:\>dir /ogn
```

(but not

```
C:\>dir /ong
```

because the G option must come first). If a directory contains more than one file with the same name, you can order the list by name and by extension within each group of names by typing

```
C:\>dir /one
```

To order the list by extension first, then by name, you would type

```
C:\>dir /oen
```

You can use a minus sign (hyphen) to reverse the effect of any option. For example, to order the list by name starting with Z instead of A, you would type

```
C:\>dir /o-n
```

To order the list by name and then by date in reverse order, with the directories at the end of the list, you would type

```
C:\>dir /o-gn-d
```

## Check out *all* your files

DOS can assign five different attributes to a file: read-only, archive, hidden, system, and directory. Read-only files can't be changed; archive files have been changed since they were last backed up to archival storage; hidden and system files are used by DOS and normally don't appear in a directory listing; and a directory file isn't a file at all—it's a subdirectory that contains other files.

Until DOS 5, the DIR command didn't show hidden and system files and otherwise ignored a file's attributes. You could use the Attribute command (ATTRIB) to see which attributes were assigned to files, and to change some of them. Starting with DOS 5, the DIR command is aware of file attributes.

The /A parameter lets you control which files are included in the list of directory entries according to their attribute, as shown in Table B.

**Table B**

| Option | Controls this attribute |
|---|---|
| /AA | Archive |
| /AR | Read-only |
| /AH | Hidden |
| /AS | System |
| /AD | Directory |

You can use the /A parameter to refer to any attribute, but the most useful option is probably D. To include only subdirectories in the directory listing, you would type

```
C:\>dir /ad
```

To include only files, you would type

```
C:\>dir /a-d
```

To include only files, in alphabetic order, you would type

```
C:\>dir /a-d /on
```

## Search those subdirectories

The /S (for *search*) parameter tells DOS to check all subdirectories when it displays the directory entries. You'll probably find this most useful for finding all files with a given name or extension on the entire drive. For example, now, when you want to clean up your disk by getting rid of all unnecessary BAK files, instead of searching through every directory that might have one, just type

```
c:\>dir c:\*.bak /s
```

## Lowercase and bare listings

A couple of other parameters let you limit the output of the DIR command to just the filename and extension (no titles or summary information, no size or date/time) or just lowercase letters. The lowercase parameter (/L) is primarily aesthetic: If you like the looks of lowercase letters better, now you can have them.

The bare (/B) filename and extension can have more use, however; you might want to include a list of filenames in a letter or other word processing document, or even in a batch file. Now you can do that just by redirecting the output of the DIR command. To create in the root directory of drive C a file named ROOTFILE.DOC containing just the name and extension of each file (but not the directories) ordered by filename, you would type

```
C:\>dir c:\ /a-d /b /on > rootfile.doc
```

## Setting your own DIR defaults

If you don't specify any parameters, the DIR command displays the directory entries of all files except system and hidden files in the order in which the files are stored on the disk. This default condition is neither selective nor especially useful, so DOS 5 lets you control the default condition of DIR with the DIRCMD environment variable.

You use the SET command to specify which parameters and options are the default condition. For example, if you'd almost always like to have DIR order the directory entries by name, you would type

```
C:\>set dircmd=/on
```

and then to view this file type

```
C:\>type rootfile.doc
```

From then on (until you turn the system off or restart DOS), typing just

```
C:\>dir
```

displays the list of directory entries ordered by filename.

If occasionally you prefer the list ordered some other way, just type the parameter with the command and it will override the default specified by DIRCMD. For example, if you have used the SET command to set the DIR default to order entries by name, as in the previous example, you can display the list ordered by size just as you would if you hadn't set a default, by typing

```
C:\>dir /os
```

If you'd like your own default to be in effect for DIR whenever you use the machine, just put the SET command in AUTOEXEC.BAT.

## Controlling DIR with a batch file

With all these choices, you might find yourself wanting to change the defaults from time to time, using one of several combinations of options that you find particularly helpful. Figure A shows a batch file called DIRMENU.BAT that displays a menu of options, asks you to choose one, carries out a SET command assigning the corresponding value to DIRCMD, then carries out the DIR command using the new set of options. In order to implement this batch file, you must create the program REPLY.COM, described in the article "Soliciting a Reply" that begins on page 4.

**Figure A**

```
@echo off
cls
echo Which type of directory?
echo.
echo 1. Files only, ordered by name, then extension
echo 2. Files only, ordered by extension, then name
echo 3. Files only, reverse ordered by size
echo 4. Directories only, ordered by name
echo 5. Restore default command form
echo.
echo Press 1-5 or Esc to cancel
:CHECK
reply
if errorlevel 54 goto CHECK
if errorlevel 53 if not errorlevel 54 set dircmd=
if errorlevel 52 if not errorlevel 53 set dircmd=/ad /on
if errorlevel 51 if not errorlevel 52 set dircmd=/a-d /o-s
if errorlevel 50 if not errorlevel 51 set dircmd=/a-d /oen
if errorlevel 49 if not errorlevel 50 set dircmd=/a-d /one
if errorlevel 27 if not errorlevel 28 goto END
if errorlevel 0 if not errorlevel 49 goto CHECK
cls
dir
:END
```

*DIRMENU.BAT creates a menu of DIR options, then carries out a SET command that corresponds to the type of directory listing you choose.*

To create DIRMENU.BAT, just load DOS 5's Editor and create a new file called DIRMENU.BAT by issuing the command

```
C:\>edit dirmenu.bat
```

Then, type the commands exactly as they appear in Figure A. Finally, save the file and exit Editor by pressing [Alt]FX and pressing *Y* in response to the save prompt.

Here's how DIRMENU.BAT works. First, the ECHO commands display the menu of options, which prompt the user to select a type of directory listing. The REPLY command runs a program that waits for you to press a key and assigns the value of the key to ERRORLEVEL so that you can check the response. The command

```
if errorlevel 54 goto CHECK
```

goes back to wait for you to press another key if you press a key whose key code is greater than the one produced by pressing 5.

The next five IF ERRORLEVEL commands check the response and carry out the SET command that corresponds to the choice you made if keys 1 through 5 were pressed. Pressing the [Esc] key sets ERRORLEVEL to 27, so the command

```
if errorlevel 27 if not errorlevel 28 goto END
```

simply skips to the end of the batch file without doing anything.

The final IF ERRORLEVEL command goes back to wait for another response if you press a key whose key code is less than the one produced by pressing 1. The only keys that cause anything to happen, then, are 1 through 5 and [Esc].

Once you've created REPLY.COM and DIRMENU.BAT, you can invoke the batch file by typing

```
C:\>dirmenu
```

When you do, the batch file will display the following menu:

```
Which type of directory?

1. Files only, ordered by name, then extension
2. Files only, ordered by extension, then name
3. Files only, reverse ordered by size
4. Directories only, ordered by name
5. Restore default command form

Press 1-5 or Esc to cancel
```

## Tailoring the batch file

You can tailor this batch file by changing the ECHO commands so they describe the combination of options you want, then changing the corresponding IF ERRORLEVEL command so that it assigns the proper combination of parameters and options to DIRCMD. Note that the IF ERRORLEVEL commands assign values to DIRCMD in the reverse order that the ECHO commands describe the options, because you must check the ERRORLEVEL starting with the largest possible value and work down.

So even though the new DOS 5 features get the most publicity, don't overlook changes to existing commands, and our old friend, the batch file. It's still possible to bend DOS to your will—or at least closer to your will—using nothing but the tools that DOS gives you. Shells and graphical user interfaces are all the rage, but there's a lot of life left in plain old unadorned DOS.

*Contributing editor Van Wolverton is the author of the best-selling books* Running MS-DOS 5 *and* Supercharging MS-DOS. *Van, who has worked for IBM and Intel, currently lives in Alberton, Montana.*

DOS FUNDAMENTALS

VERSIONS 2.1-5.0

# Understanding parent and child directories

If you're like most computer users, you've occasionally wondered why DOS automatically places the entries . and .. in a subdirectory. For instance, suppose you create a new subdirectory called \TEST by issuing the command

```
C:\>md test
```

Then, you move into that directory by typing

```
C:\>cd test
```

And finally, you list the contents of that directory by typing

```
C:\TEST>dir
```

Instead of finding the directory empty, DOS will display the following:

```
 Volume in drive C has no label
 Directory of  C:\TEST

.            <DIR>       3-15-91   11:28a
..           <DIR>       3-15-91   11:28a
        2 File(s)   11315200 bytes free

C:\TEST>_
```

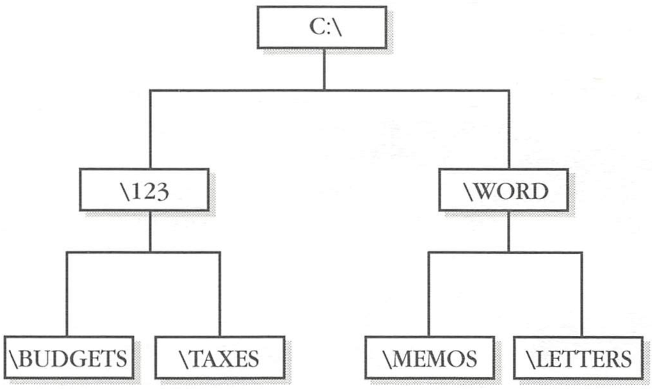
What are the two entries . and .., and why do they always appear in a subdirectory? In this article, we'll answer this question, and show you how you can use these entries to your advantage. For a related tip that takes advantage of the . entry, see the article "Two Ways to Shorten Your Command Path," which begins on page 1.

## Parent and child directories

Understanding the concept of parent and child directories is essential to understanding the . and .. entries. To illustrate this concept, consider the directory structure shown in Figure A.

As you can see, there are two directories that branch off the root directory—\123 and \WORD. These directories are called the root's *child directories*, since they branch off the root directory. In addition, the root directory is called the *parent directory* of the \123 and \WORD directories.

Figure A



*We'll use this directory structure to illustrate a few basic directory concepts.*

Just as \123 and \WORD are child directories of the root directory, \MEMOS and \LETTERS are child directories of the \WORD directory. Conversely, the \WORD directory is the parent of the \MEMOS and \LETTERS directories.

## How DOS creates a directory

As soon as you issue the MD or MKDIR command, DOS sets aside disk space to hold the entries for the directory you've created. Additionally, DOS establishes a link into and out of this directory. In other words, DOS records the address of both the new subdirectory and its parent directory, so that it can enter and leave this directory later on. This is where the . and .. entries come into play.

## The . entry

DOS uses the . (single dot) entry to store the new directory's address. Later, if you use . on the command line, DOS will refer to the address stored in the . entry to find where that directory exists on the hard disk.

Unfortunately, you can't do much with the . entry on the command line. However, you can use the . entry along with the DEL or ERASE command to erase every entry in the current subdirectory. In other words, to delete every file in the current directory, you can issue the command

```
C:\TEST>del .
```

instead of the command

```
C:\TEST>del *.*
```

## The .. entry

Just as DOS needs a . entry to record the address of each directory, it needs a .. (double dot) entry to record the address of the directory's parent. In other words, just as the . entry points to the new directory, the .. entry points one level back to the parent directory.

For instance, suppose you've set up the directory structure shown in Figure A, and you're currently working in the directory C:\123\TAXES. The .. entry in this directory points to the \TAXES subdirectory's parent directory—\123. Similarly, the .. entry in the \123 directory points to the \123 subdirectory's parent—C:\.

## Stepping through directories

You'll find that the .. entry comes in handy in a variety of situations. For instance, the .. entry makes it easier to step through the subdirectories on your hard disk. If you are in the directory C:\123\TAXES, and you want to move "back" one level to the directory's parent (C:\123), you can use the command

```
C:\123\TAXES>cd ..
C:\123>_
```

Similarly, suppose you're in the directory C:\123\TAXES, and you want to copy the file INCOME.WK1 "back" one level to the parent directory (C:\123). To do this, you can use the command

```
C:\123\TAXES>copy income.wk1 ..
```

## Conclusion

DOS automatically includes two entries (. and ..) in every subdirectory you create. In this article, we've explained the purpose of these entries. One way to take advantage of these entries is to use the . character to shorten your path. For step-by-step instructions on how to do this, see the article "Two Ways to Shorten Your Command Path" beginning on page 1.

CUSTOMIZING DOS

VERSIONS 2.1-5.0

# Fun with the DOS prompt

By David Reid

One of the nicest features of DOS's command-line interface is the configurable prompt. You don't have to settle for a simple and unimaginative prompt such as

```
C>
```

On the contrary, you can embed useful information, such as the date, time, DOS version, and current directory, into the prompt.

In this article, we'll show you how to use the PROMPT command to change DOS's command-line prompt. In the process, we'll present several sample prompts that go beyond what you might consider as typical.

## Background

If you've installed DOS 3.3 or an earlier DOS version onto a new hard disk, you've probably noticed that, unless you put a PROMPT command in the AUTOEXEC.BAT file, the system displays the

```
C>
```

prompt by default. Consequently, your next step after installing DOS was probably to add the PROMPT command

```
prompt $p$g
```

to the AUTOEXEC.BAT file so that DOS would display a more meaningful prompt, such as

```
C:\DOS>
```

Unfortunately, that's as far as many DOS users ever go when it comes to using the PROMPT command effectively. Furthermore, since DOS 5's installation program automatically adds a default PROMPT command to the AUTOEXEC.BAT file, most users simply accept the default prompt and never attempt to change it from its default value.

## Changing the DOS prompt

DOS stores the control string defining the command-line prompt in an environment variable named PROMPT. You can see the current DOS prompt definition (along with all other environment variables) by typing the SET command.

For example, on a typical DOS system, the SET command shows you something like this:

```
COMSPEC=C:\DOS\COMMAND.COM
PATH=C:\WINDOWS;C:\DOS;C:\WP
TEMP=C:\TMP
PROMPT=$p$g
```

Because DOS stores the prompt definition in an environment variable, the prompt is easy to change. As you probably know, you can change environment variables using the SET command. For example, you can change the prompt to the text string *What?* by typing

```
C:\>set prompt=What?
```

Moreover, PROMPT is one of only three environment variables you can change without using the SET command (PATH and APPEND are the others). Therefore, you can change the current DOS prompt to *What?* by typing

```
C:\>prompt What?
```

The *What?* prompt is somewhat amusing, but you'd quickly become bored with it. In fact, there aren't many static text strings that make good prompts. To create really useful prompts, you must use *metastrings*.

## What's a metastring?

What does metastring mean with respect to DOS prompts? You can think of a metastring as a substring that represents or holds the place for the actual string you place in the DOS prompt.

All metastrings for the PROMPT command are two-character sequences: the $ character followed by another character. Furthermore, there are two classes of metastrings: static and dynamic. Table A lists the static metastrings you can use in DOS prompt control strings.

**Table A**

| Metastring | Meaning |
|---|---|
| $b | ¦ (vertical bar) |
| $e | 27h (escape) |
| $g | > |
| $h | 08h (backspace) |
| $l | < |
| $q | = |
| $$ | $ |
| $_ | Carriage return/linefeed |

Incidentally, for metastrings containing letters, you may use either uppercase or lowercase—metastrings aren't case sensitive.

The static metastrings in Table A are necessary because you can't enter the actual character or string on the DOS command line without causing adverse effects. For example, you can't set the prompt to

```
COMMAND>
```

by using the command

```
C:\>prompt=COMMAND>
```

since COMMAND.COM interprets the > character as a redirection operator. The only way you can get the > character into a prompt is by using the `$g` metastring in the control string:

```
C:\>prompt=COMMAND$g
```

The static metastrings such as `$g` are useful but not very exciting. However, the dynamic metastrings listed in Table B are another story.

**Table B**

| Metastring | Meaning |
|---|---|
| $d | Current date |
| $n | Current drive |
| $p | Current drive and directory |
| $t | Current time |
| $v | DOS version number |

What's especially interesting about dynamic metastrings is you don't know what substring they will generate in the final prompt—neither does DOS for that matter. In fact, each time the previous command terminates and DOS displays a new prompt, DOS must dynamically re-evaluate the metastrings in the prompt to ensure that it puts the proper date, time, or whatever into the prompt.

Let's look at some of the prompts you can create using the PROMPT command and metastrings.

## A few example prompts

Suppose you want to change your DOS prompt to display the current time, drive, and directory. Furthermore, you want to use the symbols ==> instead of a single > symbol to point to the command line. For instance, when C:\DOS is the current directory, you want the prompt to look like

```
11:05:27  C:\DOS ==>
```

To create such a prompt, you must use the `$t` and `$p` dynamic metastrings to generate the current time and path, and the `$q` and `$g` static metastrings to generate the = and > characters. In addition, you need to add spaces between some of the metastrings.

Let's take a stab at creating the control string for this prompt. Here's our first attempt:

```
C:\>prompt $t  $p $q$q$g
```

If you type in this PROMPT command, you'll see the DOS prompt change to this format:

```
11:05:29.68  C:\DOS ==>
```

This is almost what we want, but not exactly. We need to remove from the time string the decimal point and the two digits for hundreths of seconds. We can erase these three characters from the time string by using three `$h` (backspace) metastrings:

```
C:\>prompt $t$h$h$h  $p $q$q$g
```

When you type this new PROMPT command, you'll get the desired DOS prompt with the time in HH:MM:SS form.

You might have noticed one problem with this prompt: Because of the time string, we've reduced the usable length of the command line by 10 characters. In some cases, this may be unacceptable. For example, if you're several directories deep and the current drive and directory string becomes so long that it approaches the right side of the screen, you can't enter long commands without their wrapping to the next line. You can avoid this problem by creating multi-line prompts.

Let's say you want to keep the standard `$p$g` (drive and directory) prompt, but you also want the date and time prominently displayed with each new prompt. If you display all three items on the same line, the prompt could easily extend more than halfway across the screen; this is unacceptable. The solution is to use the `$_` metastring to break the DOS prompt onto two lines.

The following PROMPT command moves down one line, displays the date and time, moves down one more line, displays the current drive and directory, and finally displays the pointer to the command line:

```
C:\>prompt $_$d  $t$h$h$h$_$p $q$q$g
```

If you type this PROMPT command, and then press [Enter] several times, you'll see a series of prompts similar to these:

```
Fri 03-15-1991  11:29:04
C:\BATCH\TEST ==>

Fri 03-15-1991  11:29:05
C:\BATCH\TEST ==>
```

As you can see, the leading `$_` metastring in the prompt definition helps separate the last line of one prompt from the first line of the next.

## Conclusion

You spend a lot of time each day staring at the DOS prompt; you deserve a nice one. As we've shown in this article, you don't have to settle for just any old prompt. You can customize the prompt to best fit your needs and tastes.

*David Reid is the editor-in-chief of The Cobb Group journals* Inside Microsoft C *and* The DOS Authority.

*Continued from page 1*

**Two ways to shorten your command path**

## Using valid drive parameters

Now that you know how the SUBST command works, you need to be aware of a few rules governing the drive parameter of this command. First of all, make sure you don't use a letter assigned to a physical drive as the drive parameter. If you do, DOS will not allow you to access that physical drive until you use another SUBST command to remove that drive letter.

By default, E is the highest drive letter DOS will recognize. If you need to use drive letters higher than E, you'll need to insert a LASTDRIVE command into your CONFIG.SYS file that takes the form

```
lastdrive=letter
```

where *letter* is the highest drive letter you want to use (don't use a colon). As you might guess, you can specify any letter up to Z for your letter parameter. If you attempt to use a drive letter in a SUBST command that is higher than the letter specified in CONFIG.SYS, DOS will display the message *Invalid parameter*.

For example, if you want to use drive letters up to Z in your SUBST commands, you would add to your CONFIG.SYS file the command:

```
lastdrive=z
```

## Listing the substitutions

You can view the drive substitutions you've made whenever you want by simply typing the command *subst* without any parameters. When you press [Enter], DOS will display a list of all the drive assignments you've made. For example, if you've previously made drive substitutions using the commands

```
C:\>subst e: c:\apps\wp\letters
C:\>subst f: c:\apps\wp\forecast
```

you can verify that DOS made these substitutions by entering the command

```
C:\>subst
```

When you press [Enter], DOS will respond

```
E: => C:\APPS\WP\LETTERS
F: => C:\APPS\WP\FORECAST
```

## Shortening your path

As we mentioned earlier, in addition to saving you keystrokes at the DOS prompt, the SUBST command lets you reduce the length of your system path. If you've installed applications in some relatively deep directories on your hard disk, you can precede the PATH command in your AUTOEXEC.BAT file with a SUBST command that assigns drive letters to those application directories. Then, you can use the new drive letter in your PATH command instead of using the longer directory names.

For example, suppose you've installed several applications in directories off the C:\APPS directory, as shown in Table A.

Table A

| You've installed... | in the directory... |
|---|---|
| Lotus 1-2-3 | C:\APPS\123 |
| WordPerfect | C:\APPS\WP |
| Harvard Graphics | C:\APPS\HG |

Now, if you want to include in your system path every application subdirectory listed in Table A as well as your DOS directory (C:\DOS), you'll want to include the following command in AUTOEXEC.BAT:

```
path c:\dos;c:\apps\123;c:\apps\wp;c:\apps\hg
```

To shorten this path, you could first use the following SUBST command to assign the drive letter K: to the directory C:\APPS:

```
c:\dos\subst k: c:\apps
```

(Note that when you specify the SUBST command in AUTOEXEC.BAT, you must precede the command with the directory that contains the SUBST.COM file.) Then, you can use the drive letter K: in place of C:\APPS everywhere in the PATH command, like this:

```
path c:\dos;k:\123;k:\wp;k:\hg
```

For this simple example, the SUBST command reduces the length of the path from 40 to 25 characters. Of course, the savings will vary according to both the number of directories you include in the path, and the length of the pathname to which you assign a drive letter.

## An even shorter path

If you want, you can create an even shorter path by assigning a different drive letter to each directory. Once you do, you can use a single character—a period—in place of the directory name in your path statement. When you include a period in your path, it tells DOS to look in the current directory of the designated drive.

Returning to our example, let's first assign a different drive letter to each subdirectory, then specify a path that's even shorter than the one shown above.

First, assign the drive letters L, M, and N to the 1-2-3, WordPerfect, and Harvard Graphics subdirectories, respectively, by issuing the following three commands:

```
c:\dos\subst l: c:\apps\123
c:\dos\subst m: c:\apps\wp
c:\dos\subst n: c:\apps\hg
```

Once you've assigned each subdirectory a separate drive letter, you can specify a substantially shorter path:

```
path l:.;m:.;n:.
```

Now you've trimed the path to a svelte 11 characters. Again, the savings you'll reap depends on the number of directories you include in the path, and the length of the pathnames.

As we mentioned earlier, the key to this technique is that by specifying a period after the drive letter, you tell DOS to look in the current directory of the specified drive. To make sure the current directory in each drive is the root directory, you might want to issue the following CD commands:

```
cd l:\
cd m:\
cd n:\
```

For more on directories and subdirectories, see the article "Understanding Parent and Child Directories" on page 8.

## One drawback

There is a drawback to using SUBST commands to assign drive letters to directories: Each substitution occupies a few bytes of memory. For this reason, make sure you don't go overboard with SUBST commands—use them to make only the substitutions you'll use on a regular basis.

## Conclusion

If you've set up multiple levels of directories on your hard disk, you'll often find yourself typing in long pathnames as you access files from DOS and from within applications. In this article, we've shown you how to use the SUBST command to eliminate the hassle of typing long pathnames and to create a substantially shorter path in your AUTOEXEC.BAT file.